Intel® Pentium® Processor Extreme Edition

Specification Update

April 2005

Notice: The Intel® Pentium® Processor Extreme Edition may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are documented in this Specification Update.

Document Number: 306832-001

Contents

Revision History

Version	Description	Date
-001	• Initial release	April 2005

§

Preface

This document is an update to the specifications contained in the documents listed in the following Affected Documents/Related Documents table. It is a compilation of device and document errata and specification clarifications and changes, and is intended for hardware system manufacturers and for software developers of applications, operating system, and tools.

Information types defined in the Nomenclature section of this document are consolidated into this update document and are no longer published in other documents. This document may also contain information that has not been previously published.

Affected Documents

Document Title	Document Number
Intel® Pentium® Processor Extreme Edition 840Δ Datasheet	306831-001

Related Documents

Document Title	Document Number
IA-32 Intel® Architecture Software Developer's Manual Volume 1: Basic Architecture, document 253665	http://developer.intel.com/design/pentium4/manuals/index_new.htm
IA-32 Intel® Architecture Software Developer's Manual Volume 2A: Instruction Set Reference Manual A–M, document 253666	http://developer.intel.com/design/pentium4/manuals/index_new.htm
IA-32 Intel® Architecture Software Developer's Manual Volume 2B: Instruction Set Reference Manual, N–Z, document 253667	http://developer.intel.com/design/pentium4/manuals/index_new.htm
IA-32 Intel® Architecture Software Developer's Manual Volume 3: System Programming Guide, document 253668	http://developer.intel.com/design/pentium4/manuals/index_new.htm
Intel® Extended Memory 64 Technology Software Developer's Guide Vol 1	http://developer.intel.com/technology/64bitextensions/300834.htm
Intel® Extended Memory 64 Technology Software Developer's Guide Vol 2	http://developer.intel.com/technology/64bitextensions/300835.htm

Nomenclature

S-Spec Number is a five-digit code used to identify products. Products are differentiated by their unique characteristics (e.g., core speed, L2 cache size, package type, etc.) as described in the processor identification information table. Care should be taken to read all notes associated with each S-Spec number

Errata are design defects or errors. Errata may cause the processor's behavior to deviate from published specifications. Hardware and software designed to be used with any given stepping must assume that all errata documented for that stepping are present on all devices.

Specification Changes are modifications to the current published specifications. These changes will be incorporated in the next release of the specifications.

Specification Clarifications describe a specification in greater detail or further highlight a specification's impact to a complex design situation. These clarifications will be incorporated in the next release of the specifications.

Documentation Changes include typos, errors, or omissions from the current published specifications. These changes will be incorporated in the next release of the specifications.

Summary Tables of Changes

The following table indicates the Specification Changes, Errata, Specification Clarifications or Documentation Changes that apply to the listed component steppings. Intel intends to fix some of the errata in a future stepping of the component, and to account for the other outstanding issues through documentation or Specification Changes as noted. This table uses the following notations:

Codes Used in Summary Table

Stepping

X:	Erratum, Specification Change or Clarification that applies to this stepping.
(No mark) or (Blank Box):	This erratum is fixed in listed stepping or specification change does not apply to listed stepping.

Status

Doc:	Document change or update that will be implemented.
PlanFix:	This erratum may be fixed in a future stepping of the product.
Fixed:	This erratum has been previously fixed.
NoFix:	There are no plans to fix this erratum.
Shaded:	This item is either new or modified from the previous version of the document.

Note: Each Specification Update item is prefixed with a capital letter to distinguish the product. The key below details the letters that are used in Intel's microprocessor Specification Updates:

A = Intel® Pentium® II processor
B = Mobile Intel® Pentium® II processor
C = Intel® Celeron® processor
D = Intel® Pentium® II Xeon™ processor
E = Intel® Pentium® III processor
F = Intel® Pentium® processor Extreme Edition
G = Intel® Pentium® III Xeon™ processor
H = Mobile Intel® Celeron® processor at 466/433/400/366/333/300 and 266 MHz
K = Mobile Intel® Pentium® III processor
L = Intel® Celeron® D processor
M = Mobile Intel® Celeron® processor

N = Intel ® Pentium® 4 processor
O = Intel ® Xeon™ processor MP
P = Intel ® Xeon™ processor
Q = Mobile Intel® Pentium® 4 processor supporting Hyper-Threading Technology on 90-nm process technology
R = Intel® Pentium® 4 processor on 90 nm process
S = 64-bit Intel® Xeon™ processor with 800 MHz system bus (1 MB and 2 MB L2 cache versions)
T = Mobile Intel® Pentium® 4 processor-M
V = Mobile Intel® Celeron® processor on .13 Micron Process in Micro-FCPGA Package
W= Intel® Celeron-M processor
X = Intel® Pentium® M processor on 90nm process with 2-MB L2 Cache
Y = Intel® Pentium® M processor
Z = Mobile Intel® Pentium® 4 processor with 533 MHz system bus

The Specification Updates for the Pentium® processor, Pentium® Pro processor, and other Intel products do not use this convention.

NO.	A0	Plan	ERRATA
F1	X	No Fix	Transaction Is Not Retried after BINIT#
F2	X	No Fix	Invalid Opcode 0FFFh Requires a ModRM Byte
F3	X	No Fix	Processor May Hang Due to Speculative Page Walks to Non-Existent System Memory
F4	X	No Fix	Memory Type of the Load Lock Different from Its Corresponding Store Unlock
F5	X	No Fix	Machine Check Architecture Error Reporting and Recovery May Not Work As Expected
F6	X	No Fix	Debug Mechanisms May Not Function as Expected
F7	X	No Fix	Cascading of Performance Counters Does Not Work Correctly When Forced Overflow Is Enabled
F8	X	No Fix	EMON Event Counting of x87 Loads May Not Work As Expected
F9	X	No Fix	System Bus Interrupt Messages without Data Which Receive a HardFailure Response May Hang the Processor
F10	X	No Fix	The Processor Signals Page-Fault Exception (#PF) Instead of Alignment Check Exception (#AC) on an Unlocked CMPXCHG8B Instruction
F11	X	No Fix	FSW May Not Be Completely Restored after Page Fault on FRSTOR or FLDENV Instructions
F12	X	No Fix	Processor Issues Inconsistent Transaction Size Attributes for Locked Operation
F13	X	No Fix	When the Processor Is in the System Management Mode (SMM), Debug Registers May Be Fully Writeable
F14	X	No Fix	Shutdown and IERR# May Result Due to a Machine Check Exception on a System with More Than One Logical Processor
F15	X	No Fix	Processor May Hang under Certain Frequencies and 12.5% STPCLK# Duty Cycle
F16	X	No Fix	System May Hang if a Fatal Cache Error Causes Bus Write Line (BWL) Transaction to Occur to the Same Cache Line Address as an Outstanding Bus Read Line (BRL) or Bus Read-Invalidate Line (BRIL}

NO.	A0	Plan	ERRATA
F17	X	No Fix	A Write to APIC Registers Sometimes May Appear to Have Not Occurred
F18	X	No Fix	Parity Error in the L1 Cache May Cause the Processor to Hang
F19	X	No Fix	Locks and SMC Detection May Cause the Processor to Temporarily Hang
F20	X	No Fix	Incorrect Debug Exception (#DB) May Occur When a Data Breakpoint is set on an FP Instruction
F21	X	No Fix	xAPIC May Not Report Some Illegal Vector Errors
F22	X	No Fix	Memory Aliasing of Pages as Uncacheable Memory Type and Write Back (WB) May Hang the System
F23	X	No Fix	Interactions Between the Instruction Translation Lookaside Buffer (ITLB) and the Instruction Streaming Buffer May Cause Unpredictable Software Behavior
F24	X[1]	No Fix	Using STPCLK# and Executing Code From Very Slow Memory Could Lead to a System Hang
F25	X	No Fix	Processor Provides a 4-Byte Store Unlock After an 8-Byte Load Lock
F26	X	No Fix	Data Breakpoints on the High Half of a Floating Point Line Split may not be Captured
F27	X	No Fix	Machine Check Exceptions May not Update Last-Exception Record MSRs (LERs)
F28	X	No Fix	MOV CR3 Performs Incorrect Reserved Bit Checking When in PAE Paging
F29	X	No Fix	Stores to Page Tables May Not Be Visible to Pagewalks for Subsequent Loads Without Serializing or Invalidating the Page Table Entry
F30	X	Plan Fix	Execution of IRET or INTn Instructions May Cause Unexpected System Behavior
F31	X	No Fix	Processor May Fault when the Upper 8 Bytes of Segment Selector is Loaded From a Far Jump Through a Call Gate via the Local Descriptor Table
F32	X	No Fix	Loading a Stack Segment with a Selector that References a Non-canonical Address can Lead to a #SS Fault on a Processor Supporting Intel® Extended Memory 64 Technology (Intel® EM64T)
F33	X	No Fix	FXRSTOR May Not Restore Non-canonical Effective Addresses on Processors with Intel® Extended Memory 64 Technology (Intel® EM64T) Enabled
F34	X	No Fix	A Push of ESP that Faults may Zero the Upper 32 Bits of RSP
F35	X	No Fix	Checking of Page Table Base Address May Not Match the Address Bit Width Supported by the Platform
F36	X	No Fix	The IA32_MCi_STATUS MSR May Improperly Indicate that Additional MCA Information May Have Been Captured
F37	X	No Fix	With TF (Trap Flag) Asserted, FP Instruction That Triggers an Unmasked FP Exception May Take Single Step Trap Before Retirement of Instruction
F38	X	No Fix	BTS(Branch Trace Store) and PEBS(Precise Event Based Sampling) May Update Memory outside the BTS/PEBS Buffer
F39	X	Plan Fix	The Base of an LDT (Local Descriptor Table) Register May be Non-zero on a Processor Supporting Intel® Extended Memory 64 Technology (Intel® EM64T)
F40	X[1]	Plan Fix	L-bit of the CS and LMA bit of the IA32_EFER Register May Have an Erroneous Value For One Instruction Following a Mode Transition in a Hyper-Threading Enabled Processor Supporting Intel® Extended Memory 64 Technology (Intel® EM64T).

NO.	A0	Plan	ERRATA
F41	X	No Fix	Memory Ordering Failure May Occur with Snoop Filtering Third Party Agents after Issuing and Completing a BWIL (Bus Write Invalidate Line) or BLW (Bus Locked Write) Transaction
F42	X	No Fix	Control Register 2 (CR2) Can be Updated during a REP MOVS/STOS Instruction with Fast Strings Enabled
F43	X	No Fix	REP STOS/MOVS Instructions with RCX ≥ 2^32 May Cause a System Hang
F44	X	Plan Fix	An REP MOVS or an REP STOS Instruction with RCX ≥ 2^32 May Fail to Execute to Completion or May Write to Incorrect Memory Locations on Processors Supporting Intel® Extended Memory 64 Technology (Intel® EM64T)
F45	X	Plan Fix	An REP LODSB or an REP LODSD or an REP LODSQ Instruction with RCX ≥ 2^32 May Cause a System Hang on Processors Supporting Intel® Extended Memory 64 Technology (Intel® EM64T)
F46	X	No Fix	A Data Access which Spans Both the Canonical and the Non-Canonical Address Space May Hang the System
F47	X[1]	Plan Fix	Running in SMM (System Management Mode) And L1 Data Cache Adaptive Mode May Cause Unexpected System Behavior when SMRAM is Mapped to Cacheable Memory
F48	X	Plan Fix	Entering Single Logical Processor Mode Via Power on Configuration Results in a System Hang at Reset
F49	X[1]	No Fix	Incorrect Duty Cycle is Chosen when On-Demand Clock Modulation is Enabled in a Processor Supporting Hyper-Threading Technology
F50	X	No Fix	Enhanced Halt State (C1E) May Not Be Entered in a System with More Than One Logical Processor
F51	X	No Fix	A 64-Bit Value of Linear Instruction Pointer (LIP) May be Reported Incorrectly in the Branch Trace Store (BTS) Memory Record or in the Precise Event Based Sampling (PEBS) Memory Record

NOTES:
1. This erratum applies only to Intel Pentium processor Extreme Edition with Hyper-Threading Technology Enabled. §

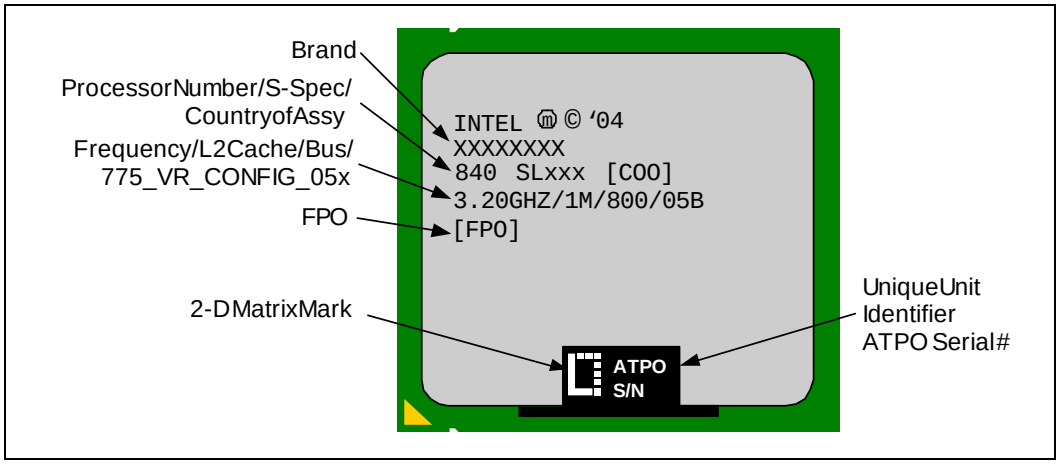
NO.	A0	Plans	SPECIFICATION CHANGES
			There are no specification changes in this Specification Update revision

NO	A0	Plans	SPECIFICATION CLARIFICATIONS
			There are no specification clarifications in this Specification Update revision

NO	A0	Plans	DOCUMENTATION CHANGES
			There are no documentation changes in this Specification Update revision

General Information

Figure 1. Intel® Pentium® Processor Extreme Edition Package



Identification Information

The Intel® Pentium® processor Extreme Edition and Intel® Pentium® D processor can be identified by the following values:

Family[1]	Model[2]
1111b	0100b

NOTES:
1. The Family corresponds to bits [11:8] of the EDX register after RESET, bits [11:8] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register, and the generation field of the Device ID register accessible through Boundary Scan.
2. The Model corresponds to bits [7:4] of the EDX register after RESET, bits [7:4] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register, and the model field of the Device ID register accessible through Boundary Scan.

Table 1. Intel Pentium Processor Extreme Edition

S-Spec	Core Stepping	L2 Cache Size (bytes)	Processor Signature	Speed Core/Bus	Package and Revision	Notes
SL8FK	A0	1M x 2	0F44h	3.20GHz/800MHz	775-land FC-LGA4 37.5 x 37.5 mm Rev 01	1, 2

NOTES:
1. These processors support the 775_VR_CONFIG_05B (performance) specifications.
2. These parts support Hyper-Threading Technology.

§

Errata

F1. Transaction Is Not Retried after BINIT#

Problem: If the first transaction of a locked sequence receives a HITM# and DEFER# during the snoop phase it should be retried and the locked sequence restarted. However, if BINIT# is also asserted during this transaction, it will not be retried.

Implication: When this erratum occurs, locked transactions will unexpectedly not be retried.

Workaround: None identified.

Status: For the steppings affected see the *Summary Tables of Changes*.

F2. Invalid Opcode 0FFFh Requires a ModRM Byte

Problem: Some invalid opcodes require a ModRM byte (or other following bytes), while others do not. The invalid opcode 0FFFh did not require a ModRM byte in previous generation Intel architecture processors, but does in the Pentium 4 processor.

Implication: The use of an invalid opcode 0FFFh without the ModRM byte may result in a page or limit fault on the Pentium 4 processor.

Workaround: Use a ModRM byte with invalid 0FFFh opcode.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F3. Processor May Hang Due to Speculative Page Walks to Non-Existent System Memory

Problem: A load operation that misses the Data Translation Lookaside Buffer (DTLB) will result in a page-walk. If the page-walk loads the Page Directory Entry (PDE) from cacheable memory and that PDE load returns data that points to a valid Page Table Entry (PTE) in uncacheable memory the processor will access the address referenced by the PTE. If the address referenced does not exist the processor will hang with no response from system memory.

Implication: Processor may hang due to speculative page walks to non-existent system memory.

Workaround: Page directories and page tables in UC memory space which are marked valid must point to physical addresses that will return a data response to the processor.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F4. Memory Type of the Load Lock Different from Its Corresponding Store Unlock

Problem: A use-once protocol is employed to ensure that the processor in a multi-agent system may access data that is loaded into its cache on a Read-for-Ownership operation at least once before it is snooped out by another agent. This protocol is necessary to avoid a multi-agent livelock scenario in which the processor cannot gain ownership of a line and modify it before that data is snooped out by another agent. In the case of this erratum, split load lock instructions incorrectly trigger the use-once protocol. A load lock operation accesses data that splits across a page boundary with both pages of WB memory type. The use-once protocol activates and the memory type for the split halves get forced to UC. Since use-once does not apply to stores, the store unlock instructions go out as WB memory type. The full sequence on the bus is: locked partial read (UC), partial read (UC), partial write (WB), locked partial write (WB). The use-once protocol should not be applied to load locks.

Implication: When this erratum occurs, the memory type of the load lock will be different than the memory type of the store unlock operation. This behavior (load locks and store unlocks having different memory types) does not introduce any functional failures such as system hangs or memory corruption.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes.*

F5. Machine Check Architecture Error Reporting and Recovery May Not Work As Expected

Problem: When the processor detects errors it should attempt to report and/or recover from the error. In the situations described below, the processor does not report and/or recover from the error(s) as intended.

- When a transaction is deferred during the snoop phase and subsequently receives a Hard Failure response, the transaction should be removed from the bus queue so that the processor may proceed. Instead, the transaction is not properly removed from the bus queue, the bus queue is blocked, and the processor will hang.
- When a hardware prefetch results in an uncorrectable tag error in the L2 cache, MC0_STATUS.UNCOR and MC0_STATUS.PCC are set but no Machine Check Exception (MCE) is signaled. No data loss or corruption occurs because the data being prefetched has not been used. If the data location with the uncorrectable tag error is subsequently accessed, an MCE will occur. However, upon this MCE, or any other subsequent MCE, .the information for that error will not be logged because MC0_STATUS.UNCOR has already been set and the MCA status registers will not contain information about the error which caused the MCE assertion but instead will contain information about the prefetch error event.
- When the reporting of errors is disabled for Machine Check Architecture (MCA) Bank 2 by setting all MC2_CTL register bits to 0, uncorrectable errors should be logged in the IA32_MC2_STATUS register but no machine-check exception should be generated. Uncorrectable loads on bank 2, which would normally be logged in the IA32_MC2_STATUS register, are not logged.
- When one-half of a 64-byte instruction fetch from the L2 cache has an uncorrectable error and the other 32-byte half of the same fetch from the L2 cache has a correctable error, the

processor will attempt to correct the correctable error but cannot proceed due to the uncorrectable error. When this occurs the processor will hang.

- When an L1 cache parity error occurs, the cache controller logic should write the physical address of the data memory location that produced that error into the IA32_MC1_ADDR REGISTER (MC1_ADDR). In some instances of a parity error on a load operation that hits the L1 cache, the cache controller logic may write the physical address from a subsequent load or store operation into the IA32_MC1_ADDR register.
- When an error exists in the tag field of a cache line such that a request for ownership (RFO) issued by the processor hits multiple tag fields in the L2 cache (the correct tag and the tag with the error) and the accessed data also has a correctable error, the processor will correctly log the multiple tag match error but will hang when attempting to execute the machine check exception handler.
- If a memory access receives a machine check error on both 64 byte halves of a 128-byte L2 cache sector, the IA32_MC0_STATUS register records this event as multiple errors, i.e., the valid error bit and the overflow error bit are both set indicating that a machine check error occurred while the results of a previous error were in the error-reporting bank. The IA32_MC1_STATUS register should also record this event as multiple errors but instead records this event as only one correctable error.
- The overflow bit should be set to indicate when more than one error has occurred. The overflow bit being set indicates that more than one error has occurred. Because of this erratum, if any further errors occur, the MCA overflow bit will not be updated, thereby incorrectly indicating only one error has been received.
- If an I/O instruction (IN, INS, REP INS, OUT, OUTS, or REP OUTS) is being executed, and if the data for this instruction becomes corrupted, the processor will signal a Machine Check Exception (MCE). If the instruction is directed at a device that is powered down, the processor may also receive an assertion of SMI#. Since MCEs have higher priority, the processor will call the MCE handler, and the SMI# assertion will remain pending. However, while attempting to execute the first instruction of the MCE handler, the SMI# will be recognized and the processor will attempt to execute the SMM handler. If the SMM handler is successfully completed, it will attempt to restart the I/O instruction, but will not have the correct machine state due to the call to the MCE handler. This can lead to failure of the restart and shutdown of the processor.
- If PWRGOOD is de-asserted during a RESET# assertion causing internal glitches, the MCA registers may latch invalid information.
- If RESET# is asserted, then de-asserted, and reasserted, before the processor has cleared the MCA registers, then the information in the MCA registers may not be reliable, regardless of the state or state transitions of PWRGOOD.
- If MCERR# is asserted by one processor and observed by another processor, the observing processor does not log the assertion of MCERR#. The Machine Check Exception (MCE) handler called upon assertion of MCERR# will not have any way to determine the cause of the MCE.
- The Overflow Error bit (bit 62) in the IA32_MC0_STATUS register indicates, when set, that a machine check error occurred while the results of a previous error were still in the error reporting bank (i.e., The Valid bit was set when the new error occurred). If an uncorrectable error is logged in the error-reporting bank and another error occurs, the overflow bit will not be set.

- The MCA Error Code field of the IA32_MC0_STATUS register gets written by a different mechanism than the rest of the register. For uncorrectable errors, the other fields in the IA32_MC0_STATUS register are only updated by the first error. Any further errors that are detected will update the MCA Error Code field without updating the rest of the register, thereby leaving the IA32_MC0_STATUS register with stale information.
- When a speculative load operation hits the L2 cache and receives a correctable error, the IA32_MC1_Status Register may be updated with incorrect information. The IA32_MC1_Status Register should not be updated for speculative loads.
- The processor should only log the address for L1 parity errors in the IA32_MC1_Status register if a valid address is available. If a valid address is not available, the Address Valid bit in the IA32_MC1_Status register should not be set. In instances where an L1 parity error occurs and the address is not available because the linear to physical address translation is not complete or an internal resource conflict has occurred, the Address Valid bit is incorrectly set.
- The processor may hang when an instruction code fetch receives a hard failure response from the system bus. This occurs because the bus control logic does not return data to the core, leaving the processor empty. IA32_MC0_STATUS MSR does indicate that a hard fail response occurred.
- The processor may hang when the following events occur and the machine check exception is enabled, CR4.MCE=1. A processor that has its STPCLK# pin asserted will internally enter the Stop Grant State and finally issue a Stop Grant Acknowledge special cycle to the bus. If an uncorrectable error is generated during the Stop Grant process it is possible for the Stop Grant special cycle to be issued to the bus before the processor vectors to the machine check handler. Once the chipset receives its last Stop Grant special cycle it is allowed to ignore any bus activity from the processors. As a result, processor accesses to the machine check handler may not be acknowledged, resulting in a processor hang.

Implication: The processor is unable to correctly report and/or recover from certain errors.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F6. Debug Mechanisms May Not Function As Expected

Problem: Certain debug mechanisms may not function as expected on the processor. The cases are as follows:

- When the following conditions occur: 1) An FLD instruction signals a stack overflow or underflow, 2) the FLD instruction splits a page-boundary or a 64 byte cache line boundary, 3) the instruction matches a Debug Register on the high page or cache line respectively, and 4) the FLD has a stack fault and a memory fault on a split access, the processor will only signal the stack fault and the debug exception will not be taken.
- When a data breakpoint is set on the ninth and/or tenth byte(s) of a floating point store using the Extended Real data type, and an unmasked floating point exception occurs on the store, the break point will not be captured.
- When any instruction has multiple debug register matches, and any one of those debug registers is enabled in DR7, all of the matches should be reported in DR6 when the processor goes to the debug handler. This is not true during a REP instruction. As an example, during

execution of a REP MOVSW instruction the first iteration a load matches DR0 and DR2 and sets DR6 as FFFF0FF5h. On a subsequent iteration of the instruction, a load matches only DR0. The DR6 register is expected to still contain FFFF0FF5h, but the processor will update DR6 to FFFF0FF1h.

- A data breakpoint that is set on a load to uncacheable memory may be ignored due to an internal segment register access conflict. In this case the system will continue to execute instructions, bypassing the intended breakpoint. Avoiding having instructions that access segment descriptor registers (e.g., LGDT, LIDT) close to the UC load, and avoiding serialized instructions before the UC load will reduce the occurrence of this erratum.

Implication: Certain debug mechanisms do not function as expected on the processor.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F7. Cascading of Performance Counters Does Not Work Correctly When Forced Overflow Is Enabled

Problem: The performance counters are organized into pairs. When the CASCADE bit of the Counter Configuration Control Register (CCCR) is set, a counter that overflows will continue to count in the other counter of the pair. The FORCE_OVF bit forces the counters to overflow on every non-zero increment. When the FORCE_OVF bit is set, the counter overflow bit will be set but the counter no longer cascades.

Implication: The performance counters do not cascade when the FORCE_OVF bit is set.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F8. EMON Event Counting of x87 Loads May Not Work As Expected

Problem: If a performance counter is set to count x87 loads and floating point exceptions are unmasked, the FPU Operand Data Pointer (FDP) may become corrupted.

Implication: When this erratum occurs, the FPU Operand Data Pointer (FDP) may become corrupted.

Workaround: This erratum will not occur with floating point exceptions masked. If floating point exceptions are unmasked, then performance counting of x87 loads should be disabled.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F9. System Bus Interrupt Messages without Data Which Receive a HardFailure Response May Hang the Processor

Problem: When a system bus agent (processor or chipset) issues an interrupt transaction without data onto the system bus and the transaction receives a HardFailure response, a potential processor hang can occur. The processor, which generates an inter-processor interrupt (IPI) that receives the HardFailure response, will still log the MCA error event cause as HardFailure, even if the APIC causes a hang. Other processors, which are true targets of the IPI, will also hang on hardfail-without-data, but will not record an MCA HardFailure event as the cause. If a HardFailure response occurs on a system bus interrupt message with data, the APIC will complete the operation so as not to hang the processor.

Implication: The processor may hang.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes.*

F10. The Processor Signals Page-Fault Exception (#PF) Instead of Alignment Check Exception (#AC) on an Unlocked CMPXCHG8B Instruction

Problem: If a Page-Fault Exception (#PF) and Alignment Check Exception (#AC) both occur for an unlocked CMPXCHG8B instruction, then #PF will be flagged.

Implication: Software that depends on the Alignment Check Exception (#AC) before the Page-Fault Exception (#PF) will be affected since #PF is signaled in this case.

Workaround: Remove the software's dependency on #AC having precedence over #PF. Alternately, correct the page fault in the page fault handler and then restart the faulting instruction

Status: For the stepping affected, see the *Summary Tables of Changes.*

F11. FSW May Not Be Completely Restored after Page Fault on FRSTOR or FLDENV Instructions

Problem: If the FPU operating environment or FPU state (operating environment and register stack) being loaded by an FLDENV or FRSTOR instruction wraps around a 64-KB or 4-GB boundary and a page fault (#PF) or segment limit fault (#GP or #SS) occurs on the instruction near the wrap boundary, the upper byte of the FPU status word (FSW) might not be restored. If the fault handler does not restart program execution at the faulting instruction, stale data may exist in the FSW.

Implication: When this erratum occurs, stale data will exist in the FSW.

Workaround: Ensure that the FPU operating environment and FPU state do not cross 64-KB or 4-GB boundaries. Alternately, ensure that the page fault handler restarts program execution at the faulting instruction after correcting the paging problem.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F12. Processor Issues Inconsistent Transaction Size Attributes for Locked Operation

Problem: When the processor is in the Page Address Extension (PAE) mode and detects the need to set the Access and/or Dirty bits in the page directory or page table entries, the processor sends an 8 byte load lock onto the system bus. A subsequent 8 byte store unlock is expected, but instead a 4 byte store unlock occurs. Correct data is provided since only the lower bytes change, however external logic monitoring the data transfer may be expecting an 8-byte store unlock.

Implication: No known commercially available chipsets are affected by this erratum.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F13. When the Processor Is in the System Management Mode (SMM), Debug Registers May Be Fully Writeable

Problem: When in System Management Mode (SMM), the processor executes code and stores data in the SMRAM space. When the processor is in this mode and writes are made to DR6 and DR7, the processor should block writes to the reserved bit locations. Due to this erratum, the processor may not block these writes. This may result in invalid data in the reserved bit locations.

Implication: Reserved bit locations within DR6 and DR7 may become invalid.

Workaround: Software may perform a read/modify/write when writing to DR6 and DR7 to ensure that the values in the reserved bits are maintained.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F14. Shutdown and IERR# May Result Due to a Machine Check Exception on a System with More Than One Logical Processor

Problem: When a Machine Check Exception (MCE) occurs due to an internal error, all logical processors normally vector to the MCE handler. However, if one of the logical processors is in the “Wait-for-SIPI” state, that logical processor will not have an MCE handler and will shut down and assert IERR#.

Implication: A system with a logical processor in the “Wait-for-SIPI” state will shut down when an MCE occurs on another logical processor.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes.*

F15. Processor May Hang under Certain Frequencies and 12.5% STPCLK# Duty Cycle

Problem: If a system de-asserts STPCLK# at a 12.5% duty cycle, the processor is running below 2 GHz, and the processor thermal control circuit (TCC) on-demand clock modulation is active, the processor may hang. This erratum does not occur under the automatic mode of the TCC.

Implication: When this erratum occurs, the processor will hang.

Workaround: If use of the on-demand mode of the processor's TCC is desired in conjunction with STPCLK# modulation, then assure that STPCLK# is not asserted at a 12.5% duty cycle.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F16. System May Hang if a Fatal Cache Error Causes Bus Write Line (BWL) Transaction to Occur to the Same Cache Line Address as an Outstanding Bus Read Line (BRL) or Bus Read-Invalidate Line (BRIL}

Problem: A processor internal cache fatal data ECC error may cause the processor to issue a BWL transaction to the same cache line address as an outstanding BRL or BRIL. As it is not typical behavior for a single processor to have a BWL and a BRL/BRIL concurrently outstanding to the same address, this may represent an unexpected scenario to system logic within the chipset.

Implication: The processor may not be able to fully execute the machine check handler in response to the fatal cache error if system logic does not ensure forward progress on the System Bus under this scenario.

Workaround: System logic should ensure completion of the outstanding transactions. Note that during recovery from a fatal data ECC error, memory image coherency of the BWL with respect to BRL/BRIL transactions is not important. Forward progress is the primary requirement.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F17. A Write to APIC Registers Sometimes May Appear to Have Not Occurred

Problem: In respect to the retirement of instructions, stores to the uncacheable memory-based APIC register space are handled in a non-synchronized way. For example if an instruction that masks the interrupt flag (e.g., CLI) is executed soon after an uncacheable write to the Task Priority Register (TPR) that lowers the APIC priority, the interrupt masking operation may take effect before the actual priority has been lowered. This may cause interrupts whose priority is lower than the initial TPR, but higher than the final TPR, to not be serviced until the interrupt flag is finally cleared (i.e., by STI instruction. Interrupts will remain pending and are not lost).

Implication: In this example the processor may allow interrupts to be accepted but may delay their service.

Workaround: This non-synchronization can be avoided by issuing an APIC register read after the APIC register write. This will force the store to the APIC register before any subsequent instructions are executed. No commercial operating system is known to be impacted by this erratum.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F18. Parity Error in the L1 Cache May Cause the Processor to Hang

Problem: If a locked operation accesses a line in the L1 cache that has a parity error, it is possible that the processor may hang while trying to evict the line.

Implication: If this erratum occurs, it may result in a system hang. Intel has not observed this erratum with any commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F19. Bus Locks and SMC Detection May Cause the Processor to Hang Temporarily

Problem: The processor may temporarily hang in an HT Technology enabled system if one logical processor executes a synchronization loop that includes one or more locks and is waiting for release by the other logical processor. If the releasing logical processor is executing instructions that are within the detection range of the self -modifying code (SMC) logic, then the processor may be locked in the synchronization loop until the arrival of an interrupt or other event.

Implication: If this erratum occurs in an HT Technology enabled system, the application may temporarily stop making forward progress. Intel has not observed this erratum with any commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F20. Incorrect Debug Exception (#DB) May Occur When a Data Breakpoint Is Set on an FP Instruction

Problem: The default Microcode Floating Point Event Handler routine executes a series of loads to obtain data about the FP instruction that is causing the FP event. If a data breakpoint is set on the instruction causing the FP event, the load in the microcode routine will trigger the data breakpoint resulting in a Debug Exception.

Implication: An incorrect Debug Exception (#DB) may occur if data breakpoint is placed on an FP instruction. Intel has not observed this erratum with any commercially available software or system.

Workaround: None identified.

Status: For the steppings affected, see the Summary *Tables of Changes*.

F21. xAPIC May Not Report Some Illegal Vector Errors

Problem: The local xAPIC has an Error Status Register, which records all errors. The bit 6 (the Receive Illegal Vector bit) of this register, is set when the local xAPIC detects an illegal vector in a received message. When an illegal vector error is received on the same internal clock that the error status register is being written (due to a previous error), bit 6 does not get set and illegal vector errors are not flagged

Implication: The xAPIC may not report some Illegal Vector errors when they occur at approximately the same time as other xAPIC errors. The other xAPIC errors will continue to be reported.

Workaround: None identified

Status: For the stepping affected, see the *Summary Tables of Changes.*

F22. Memory Aliasing of Pages As Uncacheable Memory Type and Write Back (WB) May Hang the System

Problem: When a page is being accessed as either Uncacheable (UC) or Write Combining (WC) and WB, under certain bus and memory timing conditions, the system may loop in a continual sequence of UC fetch, implicit writeback, and Request for Ownership (RFO) retries.

Implication: This erratum has not been observed in any commercially available operating system or application. The aliasing of memory regions, a condition necessary for this erratum to occur, is documented as being unsupported in the *IA-32 Intel® Architecture Software Developer's Manual*, Volume 3, section 10.12.4, Programming the PAT. However, if this erratum occurs, the system may hang.

Workaround: The pages should not be mapped as either UC or WC and WB at the same time.

Status: For the stepping affected, see the *Summary Tables of Changes.*

F23. Interactions between the Instruction Translation Lookaside Buffer (ITLB) and the Instruction Streaming Buffer May Cause Unpredictable Software Behavior

Problem: Complex interactions within the instruction fetch/decode unit may make it possible for the processor to execute instructions from an internal streaming buffer containing stale or incorrect information.

Implication: When this erratum occurs, an incorrect instruction stream may be executed resulting in unpredictable software behavior.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the stepping affected, see the *Summary Tables of Changes.*

F24. Using STPCLK# and Executing Code from Very Slow Memory Could Lead to a System Hang

Problem: The system may hang when the following conditions are met:

1. Periodic STPCLK# mechanism is enabled via the chipset
2. Hyper-Threading Technology is enabled
3. One logical processor is waiting for an event (i.e. hardware interrupt)
4. The other logical processor executes code from very slow memory such that every code fetch is deferred long enough for the STPCLK to be re-asserted.

Implication: If this erratum occurs, the processor will go into and out of the sleep state without making forward progress, since the logical processor will not be able to service any pending event. This erratum has not been observed in any commercial platform running commercial software.

Workaround: None

Status: For the steppings affected, see the *Summary Tables of Changes.*

F25. Processor Provides a 4-Byte Store Unlock after an 8-Byte Load Lock

Problem: When the processor is in the Page Address Extension (PAE) mode and detects the need to set the Access and/or Dirty bits in the page directory or page table entries, the processor sends an 8 byte load lock onto the system bus. A subsequent 8 byte store unlock is expected, but instead a 4 byte store unlock occurs. Correct data is provided since only the lower bytes change, however external logic monitoring the data transfer may be expecting an 8 byte load lock.

Implication: No known commercially available chipsets are affected by this erratum.

Workaround: None identified at this time.

Status: For the steppings affected, see the *Summary Tables of Changes.*

F26. Data Breakpoints on the High Half of a Floating Point Line Split May Not Be Captured

Problem: When a floating point load which splits a 64-byte cache line gets a floating point stack fault, and a data breakpoint register maps to the high line of the floating point load, internal boundary conditions exist that may prevent the data breakpoint from being captured.

Implication: When this erratum occurs, a data breakpoint will not be captured.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F27. Machine Check Exceptions May not Update Last-Exception Record MSRs (LERs)

Problem: The Last-Exception Record MSRs (LERs) may not get updated when Machine Check Exceptions occur.

Implication: When this erratum occurs, the LER may not contain information relating to the machine check exception. They will contain information relating to the exception prior to the machine check exception.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F28. MOV CR3 Performs Incorrect Reserved Bit Checking When in PAE Paging

Problem: The MOV CR3 instruction should perform reserved bit checking on the upper unimplemented address bits. This checking range should match the address width reported by CPUID instruction 0x8000008. This erratum applies whenever PAE is enabled.

Implication: Software that sets the upper address bits on a MOV CR3 instruction and expects a fault may fail. This erratum has not been observed with commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F29. Stores to Page Tables May Not Be Visible to Pagewalks for Subsequent Loads without Serializing or Invalidating the Page Table Entry

Problem: Under rare timing circumstances, a page table load on behalf of a programmatically younger memory access may not get data from a programmatically older store to the page table entry if there is not a fencing operation or page translation invalidate operation between the store and the younger memory access. Refer to the IA-32 Intel® Architecture Software Developer's Manual for the correct way to update page tables. Software that conforms to the Software Developer's Manual will operate correctly.

Implication: If the guidelines in the Software Developer's Manual are not followed, stale data may be loaded into the processor's Translation Lookaside Buffer (TLB) and used for memory operations. This erratum has not been observed with any commercially available software.

Workaround: The guidelines in the IA-32 Intel® Architecture Software Developer's Manual should be followed.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F30. Execution of IRET or INTn Instructions May Cause Unexpected System Behavior

Problem: There is a small window of time, requiring alignment of many internal micro architectural events, during which the speculative execution of the IRET or INTn instructions in protected or IA-32e mode may result in unexpected software or system behavior.

Implication: This erratum may result in unexpected instruction execution, events, interrupts or a system hang when the IRET instruction is executed. The execution of the INTn instruction may cause debug breakpoints to be missed.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F31. Processor May Fault When the Upper 8 Bytes of Segment Selector Is Loaded from a Far Jump through a Call Gate via the Local Descriptor Table

Problem: In IA-32e mode of the Intel EM64T processor, control transfers through a call gate via the Local Descriptor Table (LDT) that uses a 16-byte descriptor, the upper 8-byte access may wrap and access an incorrect descriptor in the LDT. This only occurs on an LDT with a LIMIT>0x10008 with a 16-byte descriptor that has a selector of 0xFFFC.

Implication: In the event this erratum occurs, the upper 8-byte access may wrap and access an incorrect descriptor within the LDT, potentially resulting in a fault or system hang. Intel has not observed this erratum with any commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F32. Loading a Stack Segment with a Selector that References a Non-canonical Address Can Lead to a #SS Fault on a Processor Supporting Intel® Extended Memory 64 Technology (Intel® EM64T)

Problem: When a processor supporting Intel EM64T is in IA-32e mode, loading a stack segment with a selector which references a non-canonical address will result in a #SS fault instead of a #GP fault.

Implication: When this erratum occurs, Intel EM64T enabled systems may encounter unexpected behavior.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F33. FXRSTOR May Not Restore Non-canonical Effective Addresses on Processors with Intel® Extended Memory 64 Technology (Intel® EM64T) Enabled

Problem: If an x87 data instruction has been executed with a non-canonical effective address, FXSAVE may store that non-canonical FP Data Pointer (FDP) value into the save image. An FXRSTOR instruction executed with 64-bit operand size may signal a General Protection Fault (#GP) if the FDP or FP Instruction Pointer (FIP) is in non-canonical form.

Implication: When this erratum occurs, Intel EM64T enabled systems may encounter an unintended #GP fault.

Workaround: Software should avoid using non-canonical effective addressing in EM64T enabled processors. BIOS can contain a workaround for this erratum removing the unintended #GP fault on FXRSTOR.

Status: For the steppings affected, see the Summary Tables of Changes.

F34. A Push of ESP That Faults May Zero the Upper 32 Bits of RSP

Problem: In the event that a push ESP instruction, that faults, is executed in compatibility mode, the processor will incorrectly zero upper 32-bits of RSP.

Implication: A Push of ESP in compatibility mode will zero the upper 32-bits of RSP. Due to this erratum, this instruction fault may change the contents of RSP. This erratum has not been observed in commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F35. Checking of Page Table Base Address May Not Match the Address Bit Width Supported by the Platform

Problem: If the page table base address, included in the page map level-4 table, page-directory pointer table, page-directory table or page table, exceeds the physical address range supported by the platform (e.g., 36-bit) and it is less than the implemented address range (e.g., 40-bit), the processor does not check if the address is invalid.

Implication: If software sets such invalid physical address in those tables, the processor does not generate a page fault (#PF) upon access to that virtual address, and the access results in an incorrect read or write. If BIOS provides only valid physical address ranges to the operating system, this erratum will not occur.

Workaround: BIOS must provide valid physical address ranges to the operating system.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F36. The IA32_MCi_STATUS MSR May Improperly Indicate that Additional MCA Information May Have Been Captured

Problem: When a data parity error is detected and the bus queue is busy, the ADDRV and MISCV bits of the IA32_MCi_STATUS register may be asserted even though the contents of the IA32_MCi_ADDR and IA32_MCi_MISC MSRs were not properly captured.

Implication: If this erratum occurs, the MCA information captured in the IA32_MCi_ADDR and IA32_MCi_MISC may not correspond to the reported machine-check error, even though the ADDRV and MISCV are asserted.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F37. With TF (Trap Flag) Asserted, FP Instruction That Triggers an Unmasked FP Exception May Take Single Step Trap before Retirement of Instruction

Problem: If an FP instruction generates an unmasked exception with the EFLAGS.TF=1, it is possible for external events to occur, including a transition to a lower power state. When resuming from the lower power state, it may be possible to take the single step trap before the execution of the original FP instruction completes.

Implication: A Single Step trap will be taken when not expected.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F38. BTS(Branch Trace Store) and PEBS(Precise Event Based Sampling) May Update Memory outside the BTS/PEBS Buffer

Problem: If the BTS/PEBS buffer is defined such that:

- The difference between BTS/PEBS buffer base and BTS/PEBS absolute maximum is not an integer multiple of the corresponding record sizes
- BTS/PEBS absolute maximum is less than a record size from the end of the virtual address space
- The record that would cross BTS/PEBS absolute maximum will also continue past the end of the virtual address space

A BTS/PEBS record can be written that will wrap at the 4G boundary (IA32) or 2^64 boundary (EM64T mode), and write memory outside of the BTS/PEBS buffer.

Implication: Software that uses BTS/PEBS near the 4G boundary (IA32) or 2^64 boundary (EM64T mode), and defines the buffer such that it does not hold an integer multiple of records can update memory outside the BTS/PEBS buffer.

Workaround: Define BTS/PEBS buffer such that BTS/PEBS absolute maximum minus BTS/PEBS buffer base is integer multiple of the corresponding record sizes as recommended in the IA-32 Intel® Architecture Software Developer's Manual, Volume 3.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F39. The Base of an LDT (Local Descriptor Table) Register May be Non-zero on a Processor Supporting Intel® Extended Memory 64 Technology (Intel® EM64T)

Problem: In IA-32e mode of an Intel EM64T-enabled processor, the base of an LDT register may be non-zero.

Implication: Due to this erratum, Intel EM64T-enabled systems may encounter unexpected behavior when accessing an LDT register using the null selector. There may be no #GP fault in response to this access.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F40. L-bit of the CS and LMA bit of the IA32_EFER Register May Have an Erroneous Value For One Instruction Following a Mode Transition in a Hyper-Threading Enabled Processor Supporting Intel® Extended Memory 64 Technology (Intel® EM64T)

Problem: In an Intel® EM64T enabled Processor, the L-bit of the Code Segment (CS) descriptor may not update with the correct value in an HT environment. This may occur in a small window when one logical processor is making a transition from compatibility mode to 64-bit mode (or vice-versa) while the other logical processor is being stalled. A similar problem may occur for the observation of the EFER.LMA bit by the decode logic.

Implication: The first instruction following a mode transition may be decoded as if it was still in the previous mode. For example, this may result in an incorrect stack size used for a stack operation, i.e. a write of only 4-bytes and an adjustment to ESP of only 4 in 64-bit mode. The problem can manifest itself; however, on any instruction that would behave differently in 64-bit mode than in compatibility mode.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F41. Memory Ordering Failure May Occur with Snoop Filtering Third Party Agents after Issuing and Completing a BWIL (Bus Write Invalidate Line) or BLW (Bus Locked Write) Transaction

Problem: Under limited circumstances, the processors may, after issuing and completing a BWIL or BLW transaction, retain data from the addressed cache line in shared state even though the specification requires complete invalidation. This data retention may also occur when a BWIL transaction's self-snooping yields HITM snoop results.

Implication: A system may suffer memory ordering failures if its central agent incorporates coherence sequencing which depends on full self-invalidation of the cache line associated (1) with BWIL and BLW transactions, or (2) all HITM snoop results without regard to the transaction type and snoop results source.

Workaround: 1. The central agent can issue a bus cycle that causes a cache line to be invalidated (Bus Read Invalidate Line (BRIL) or BWIL transaction) in response to a processor-generated BWIL (or BLW) transaction to insure complete invalidation of the associated cache line. If there are no intervening processor-originated transactions to that cache line, the central agent's invalidating snoop will get a clean snoop result.

Or

2. Snoop filtering central agents can:

a. Not use processor-originated BWIL or BLW transactions to update their snoop filter information, or

b. Update the associated cache line state information to shared state on the originating bus (rather than invalid state) in reaction to a BWIL or BLW.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F42. Control Register 2 (CR2) Can be Updated during a REP MOVS/STOS Instruction with Fast Strings Enabled

Problem: Under limited circumstances while executing a REP MOVS/STOS string instruction, with fast strings enabled, it is possible for the value in CR2 to be changed as a result of an interim paging event, normally invisible to the user. Any higher priority architectural event that arrives and is handled while the interim paging event is occurring may see the modified value of CR2.

Implication: The value in CR2 is correct at the time that an architectural page fault is signaled. Intel has not observed this erratum with any commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F43. REP STOS/MOVS Instructions with RCX ≥ 2^32 May Cause a System Hang

Problem: In IA-32e mode using Intel EM64T-enabled processors, executing a repeating string instruction with the iteration count greater than or equal to 2^32 and a pending event may cause the REP STOS/MOVS instruction to live lock and hang.

Implication: When this erratum occurs, the processor may live lock and result in a system hang. Intel has not observed this erratum with any commercially available software.

Workaround: Do not use strings larger than 4 GB.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F44. An REP MOVS or an REP STOS Instruction with RCX ≥ 2^32 May Fail to Execute to Completion or May Write to Incorrect Memory Locations on Processors Supporting Intel® Extended Memory 64 Technology (Intel® EM64T)

Problem: In IA-32e mode using Intel EM64T-enabled processors, an REP MOVS or an REP STOS instruction executed with the register RCX ≥ 2^32, may fail to execute to completion or may write data to incorrect memory locations.

Implication: This erratum may cause an incomplete instruction execution or incorrect data in the memory. Intel has not observed this erratum with any commercially available software.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F45. An REP LODSB or an REP LODSD or an REP LODSQ Instruction with RCX ≥ 2^32 May Cause a System Hang on Processors Supporting Intel® Extended Memory 64 Technology (Intel® EM64T)

Problem: In IA-32e mode using Intel EM64T-enabled processors, an REP LODSB or an REP LODSD or an REP LODSQ instruction executed with the register RCX ≥ 2^32 may fail to complete execution causing a system hang. Additionally, there may be no #GP fault due to the non-canonical address in the RSI register.

Implication: This erratum may cause a system hang on Intel EM64T-enabled platforms. Intel has not observed this erratum with any commercially available software.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F46. A Data Access which Spans Both the Canonical and the Non-Canonical Address Space May Hang the System

Problem: If a data access causes a page split across the canonical to non-canonical address space the processor may livelock which in turn would cause a system hang.

Implication: When this erratum occurs, the processor may livelock, resulting in a system hang. Intel has not observed this erratum with any commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F47. Running in SMM (System Management Mode) And L1 Data Cache Adaptive Mode May Cause Unexpected System Behavior when SMRAM is Mapped to Cacheable Memory

Problem: In a Hyper-Threading Technology-enabled system, unexpected system behavior may occur if a change is made to the value of the CR3 result from an RSM (Resume From System Management) instruction while in L1 data cache adaptive mode (IA32_MISC_ENABLES MSR 0x1a0, bit 24). This behavior will only be visible when SMRAM is mapped into WB/WT cacheable memory on SMM entry and exit.

Implication: This erratum can have multiple failure symptoms including incorrect data in memory. Intel has not observed this erratum with any commercially available software.

Workaround: Disable L1 data cache adaptive mode by setting the L1 Data Cache Context Mode control (bit 24) of the IA32_MISC_ENABLES MSR (0x1a0) to 1.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F48. Entering Single Logical Processor Mode Via Power on Configuration Results in a System Hang at Reset

Problem: When the system uses power on configuration (POC) to enter single logical processor mode on a dual core processor (by asserting A31# at the deassertion of RESET#), the system will hang at reset.

Implication: POC can not be used to enter single logical processor mode.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F49. Incorrect Duty Cycle is Chosen when On-Demand Clock Modulation Is Enabled in a Processor Supporting Hyper-Threading Technology

Problem: When a processor supporting Hyper-Threading Technology enables On-Demand Clock Modulation on both logical processors, the processor is expected to select the lowest duty cycle of the two potentially different values. When one logical processor enters the AUTOHALT state, the duty cycle implemented should be unaffected by the halted logical processor. Due to this erratum, the duty cycle is incorrectly chosen to be the higher duty cycle of both logical processors.

Implication: Due to this erratum, higher duty cycle may be chosen when the On-Demand Clock Modulation is enabled on both logical processors.

Workaround: None identified at this time

Status: For the steppings affected, see the *Summary Tables of Changes*.

F50. Enhanced Halt State (C1E) May Not Be Entered in a System with More Than One Logical Processor

Problem: If the IA32_MISC_ENABLE MSR (0x1A0) C1E enable bit is not set prior to an INIT event on a system with more than one logical processor, the processor will not enter C1E until the next SIPI wakeup event for the logical processor in the "Wait-for-SIPI" state.

Implication: Due to this erratum, the processor will not enter C1E state.

Workaround: If C1E is supported in the system, the IA32_MISC_ENABLE MSR should be enabled prior to issuing the first SIPI to another logical processor.

Status: For the steppings affected, see the *Summary Tables of Changes*.

F51. A 64-Bit Value of Linear Instruction Pointer (LIP) May be Reported Incorrectly in the Branch Trace Store (BTS) Memory Record or in the Precise Event Based Sampling (PEBS) Memory Record

Problem: On a processor supporting Intel® EM64T,

- If an instruction fetch wraps around the 4G boundary in Compatibility Mode, the 64-bit value of LIP in the BTS memory record will be incorrect (upper 32 bits will be set to FFFFFFFFh when they should be 0).
- If a PEBS event occurs on an instruction whose last byte is at memory location FFFFFFFFh, the 64-bit value of LIP in the PEBS record will be incorrect (upper 32 bits will be set to FFFFFFFFh when they should be 0).

Implication: Intel has not observed this erratum on any commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

§

Specification Changes

The Specification Changes listed in this section apply to the following documents:

- *Intel® Pentium® Processor Extreme Edition 840 Datasheet*

All Specification Changes will be incorporated into a future version of the appropriate Pentium processor Extreme Edition documentation.

There are no specification changes in this Specification Update revision.

§

Specification Clarifications

The Specification Clarifications listed in this section apply to the following documents:

- *Intel® Pentium® Processor Extreme Edition 840 Datasheet*

All Specification Clarifications will be incorporated into a future version of the Pentium processor Extreme Edition documentation.

There are no specification clarifications in this Specification Update revision.

§

Documentation Changes

The Documentation Changes listed in this section apply to the following documents:

- *Intel® Pentium® Processor Extreme Edition 840[4] Datasheet*

All Documentation Changes will be incorporated into a future version of the appropriate Pentium 4 processor documentation.

Note: Documentation changes for IA-32 Intel® Architecture Software Developer's Manual volumes 1, 2A, 2B, 3 and Intel® Extended Memory 64 Technology Software Developer's Guide volumes 1, 2 will be posted in a separate document *IA-32 Intel® Architecture and Intel® Extended Memory 64 Technology Software Developer's Manual Documentation Changes*. Follow the link below to become familiar with this file.

http://developer.intel.com/design/pentium4/specupdt/252046.htm

There are no documentation changes in this Specification Update revision.

§